Java Data Objects

Jan Västernäs

Callista Enterprise AB

http://www.callista.se/enterprise

CALLISTA

Agenda

- Persistence solutions
- JDO general
- Demo
- JDO specifics
- Comparison
- Outstanding questions

Different solutions to the Persistence problem

- Produce your own code
  - By hand
  - By Code Generator
- CMP Entity Beans
- Persistence framework
  - Your own
  - Hibernate
  - JDO
  - Other . . . .

CALLISTA

JDO Resources

- Specification
- Sun Reference implementation
- Vendor implementations
  - KODO by Solarmetrics
  - TJDO open source
- JDOCentral.com
- Java Data Objects
  - Craig Russel
    - JDO Spec Lead
    - CMP Architect Sun One Server

CALLISTA

JDO Primary Objective

”To provide a transparent Java-centric view of persistent information stored in a wide variety of datastores”

Transparent

- Use POJO's to represent information
  - Plain Ordinary Java Objects
- Add persistence capabilities later

CALLISTA

Java-centric

- POJO's
- References, Collections , inheritance etc
- Traverse the model

Wide variety of datastores

- Relational Databases
- Object Database
- File-system
- Other protocol

Using JDO

- Write POJO's
- Create properties file/environment/other
- Enhance them
- Create .jdo file listing persistent classes (incl. metadata)
- Add calls to a Persistence Manager (not completely transparent)
- Create mapping to datastore
- Create datastore
- Add some jars/product
- We are in business

Hello World demo

- 7 minutes

Write objects

- Getters/setters
- References, single or multiple
- Inheritance

Enhance them

- Add persistance capabilities
- A lot of attributes and methods are added to the class
- Easy if you use ant to build
- Requires support if you use an IDE
  - KODO Eclips plugin

.jdo File

- Per package
- Describes all persistent classes
- Attributes does not have to be described, all non-transient attributes are made persistant
- Primary key must be specified
  - If not jdo creates identity field

Add Persistance Manager calls

- Start transaction
- Commit/rollback transaction
- Make Persistent (new Objects)
- Delete Objects
- Run queries
  - Find by identity (Primary key)
  - Find all objects (Entity iterator)
  - Find by custom Query

CALLISTA

Mapping to datastore

- Vendor specific
- myapp.mapping
  - Maps each java class to table and columns names (RDBMS)
- Myapp.schema
  - Contains datastore-specific information like column types (varchar, integer etc)

Create datastore

- Datastore type specific
- Vendor specific
- KODO has tool that generates DDL file

More JDO details

- Create, update, delete
- Read
- Queries
- Relations
- Identity
- Object Caching
- Optimistic transactions

IDE Integration

- Kodo offers plugins for many popular products
- Eclips plugin takes care of enhance after compile
- Also generates schema and mapping

CALLISTA

JDO modes

- NonManaged
  - JDO DB connection incl pooling
  - JDO prepared statement cache etc
  - JDO transaction demarcation
- Managed (Appserver environment)
  - Access to PersistentManager by JCA (recommended) or JNDI
  - Enroll with Container Transaction
  - Use Container-provided DataSource
  - Requires code-changes compared to NonManaged code
    - remove tx begin() and commit() calls
    - getPM-implementation

JDO in a J2EE application

- Persisence layer
- Session Bean infront
- JDO-specific DAO implementation
- Non-JDO DTO:s in DAO interface

JDO compared to CMP

- POJO based
- Supports inheritance
- Supports more relation datatypes List, Array, Map
- Different type of Query Language
- Non-transactional read
- More functionality and options (product specific)
  - Kodo: honor RI constraints
- Trickier
- Can execute without App-server
- Remote invocation not supported (god !!)

Hibernate vs JDO and CMP

- Hibernate rejects the use of build-time code generation / bytecode processing. Instead, reflection and runtime bytecode generation are used and SQL generation occurs at system startup time. This decision ensures that Hibernate does not impact upon IDE debugging and incremental compile.
- POJO based
- XML Metadata
- App server integration (J2A) in progress

Open questions JDO

- Appserver integration
  - Risk : multiple vendor products must work tight together
- DB updates at end of transaction
- How good is caching in a OLTP read/write environment
  - Flush before query
- POJO layer penetration in a service-oriented architecture
- Specification vs product specific features
  - Portability issues

CALLISTA